NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2000; **1**:1 *Prepared using* nlaauth.cls *[Version: 2000/03/22 v1.0]*

Preconditioning strategies for linear systems arising in tire design

Maria Sosonkina[1,*], John T. Melson[2], Yousef Saad[3,] and Layne T. Watson[4]

[1] *Department of Computer Science, University of Minnesota, Duluth, 320 Heller Hall, 10 University Drive, Duluth, Minnesota 55812-2496. masha@d.umn.edu*
[2] *Michelin Americas Research & Development Corp., 515 Michelin Road, P.O. Box 1987, Greenville, SC 29602-1987*
[3] *Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455. saad@cs.umn.edu*
[4] *Departments of Computer Science and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0106. ltw@cs.vt.edu*

SUMMARY

This paper discusses the application of iterative methods for solving linear systems arising in static tire equilibrium computation. The heterogeneous material properties, nonlinear constraints, and a three dimensional finite element formulation make the linear systems arising in tire design difficult to solve by iterative methods. An analysis of the matrix characteristics helps understand this behavior. This paper focuses on two preconditioning techniques: a variation of an incomplete LU factorization with threshold and a multilevel recursive solver. We propose to adapt these techniques in a number of ways to work for a class of realistic applications. In particular, it was found that these preconditioners improve convergence *only* when a rather large shift value is added to the matrix diagonal. A combination of other techniques such as filtering of small entries, pivoting in preconditioning, and a special way of defining levels for the multilevel recursive solver are shown to make these preconditioning strategies efficient for problems in tire design. We compare these techniques and assess their applicability when the linear system difficulty varies for the same class of problems. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: incomplete LU factorization, multilevel preconditioning, ill-conditioned linear systems, generalized minimum residual method

1. Introduction

Static equilibrium computation routinely takes place in the tire manufacturing process. Tire stability analysis is based on a 3D finite element model with distributed loads (see, e.g., [1]). Due to the elongated shape of the tire, there is a large difference in the numbers of elements

*Correspondence to: Department of Computer Science, University of Minnesota, Duluth, 320 Heller Hall, 10 University Drive, Duluth, Minnesota 55812-2496. masha@d.umn.edu

Contract/grant sponsor: This work is supported by Michelin Americas Research and Development Corporation and in part by the Minnesota Supercomputer Institute

Received December 1999
Accepted June 2000

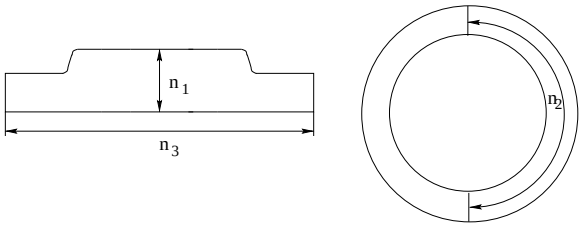

Figure 1. Three dimensional topology of the tire model

n_1, n_2, and n_3 modeling height, length, and width of the tire, respectively (Figure 1). Here we consider a medium tire model $\mathcal{M}$ and a large tire model $\mathcal{L}$. For the model $\mathcal{M}$, the total number of degrees of freedom is $n = 49800$ and the number of nonzeros is approximately $84n$. For the model $\mathcal{L}$, n is 84180 and there are approximately $86n$ nonzeros. A characteristic feature of both models is that they incorporate orthotropic and isotropic as well as heterogeneous material properties, which reflect the layered design of tires.

Computation of static equilibrium involves minimizing the potential energy $\Pi(u)$ with respect to finite element nodal displacements $u^i (i = 1, 2, 3)$ subject to nonlinear boundary conditions, which change the symmetry of a tire. The equilibrium equations of the model are obtained by setting the variation $\delta\Pi(u)$ to zero, or equivalently

$$\nabla \Pi(u) = 0.$$

The Jacobian matrix of the equilibrium equations is obtained by finite difference approximations. The distributed load is scaled by a (loading) parameter λ, and as λ varies the static equilibrium solutions trace out a curve. The difficulty of the finite element problems and concomitant linear systems varies considerably along this equilibrium curve, as well as within the nonlinear iterations to compute a particular point on this curve.

A recent preliminary study of iterative methods applied to this class of problems in tire design is reported in [15]. It was found that a variation of the incomplete LU factorization with pivoting serves well as a preconditioner to achieve acceptable speed of convergence for the model $\mathcal{M}$. This paper presents results from applying a multilevel preconditioning scheme [13] and considers larger (model $\mathcal{L}$) problems.

The rest of the paper is organized as follows. Section 2 discusses the matrix characteristics. The choice of accelerator and preconditioning strategies is explained in Section 3. A comparison of the two preconditioning strategies is provided in the numerical experiments section (Section 4). Section 5 summarizes the results and suggests future work.

2. Matrix characteristics

In nonlinear stability analysis, a sequence of sparse linear systems must be solved. Although the systems in a sequence have many (mostly structural) characteristics in common, their numerical properties vary substantially. The structural symmetry of the resulting matrices is preserved throughout the equilibrium computation. Each matrix has a 3×3 block structure, in which some of the entries in the block may have zero values. The matrix block structure

Figure 2. Matrix pattern and values before reordering.

can be readily exploited in the solution process by treating a block as a single entry. Such techniques are quite common in computational fluid dynamics applications, see for example [4]. The matrices have been reordered for the linear system solution. Sparse matrix reordering plays an important role in *any* solution technique applied. For direct methods that are based on complete matrix factorization, it governs the work and storage required. For iterative methods, it affects the quality of the preconditioning. Figure 2 shows the original element numbering, which is disastrous for either a direct or iterative solution technique: for a direct method fill-in is excessive and pivoting is required, while a preconditioner for an iterative method fails because of zeros on the diagonal and also needs pivoting. Figure 3 depicts the matrix reordered according to the element connectivity to minimize the average nonzero bandwidth. For the model $\mathcal{M}$, the resulting maximum and average bandwidths are 3840 and 1920, respectively, with 90% of the nonzeros in the band of width 2645 as reported by a routine from [11]. Applying a direct method to the reordered matrix 90% of the bandwidth becomes full.

Due to the presence of extremely heterogeneous material properties, the relative magnitudes of the entries in the resulting stiffness matrix differ greatly (Figures 2 and 3). A result is that

many diagonal entries tend to be relatively small. The row i of matrix $A = (a_{ij})$ is weakly diagonally dominant if

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|.$$

A large number of nondiagonally-dominant rows and columns is typical of the matrices in a sequence. The percent of (weakly) diagonally dominant rows ranges between 3.5% and 6%. The smaller this number, the more difficult the linear systems considered here are to solve iteratively. Near the beginning of the equilibrium computation, the matrices are symmetric in value and have more (weakly) diagonally dominant rows. Near the end of the equilibrium computation the linear systems have pronounced asymmetries due to the treatment of stationary solutions of rotation [8]. For example, a matrix of the linear system in the model $\mathcal{M}$ towards the end of the nonlinear convergence has the Frobenius norm of its nonsymmetric part $(A - A^t)/2$ equal to 1.84×10^7 with a Frobenius norm of 4.38×10^{11} for the whole matrix A.

The spectrum of the matrix can sometimes be quite informative about difficulties iterative methods might encounter. Given the size of the matrix, it is not practically feasible to compute all its eigenvalues. In fact computing the whole spectrum would not be useful, but a look at the eigenvalue estimates nearest to the origin shows a very strong clustering around zero. This gives another sign of potential trouble for iterative methods. As an example, Figure 4 shows the 200 leftmost eigenvalue estimates obtained for the matrix $\mathcal{M}$. The eigenvalues are not computed accurately. Their estimates result from the use of the deflated GMRES algorithm described in the next section. However, the strong clustering is certainly real.

In addition to extremely unfavorable clustering of the spectrum, the matrices tend also to be very ill-conditioned. Thus, a typical 1-norm condition number for the medium model problem was calculated to be approximately $\kappa_1 \approx 1.69$ e+10. This is obtained from the sparse direct factorization code SuperLU [6]. More than the pure condition number, the clustering of eigenvalues around zero may be the most damaging of all the characteristics of the matrix.

3. Accelerators and preconditioners considered

For linear systems that are hard to solve by iterative methods, it is important to combine enhanced accelerators with accurate and efficient preconditioners. For systems arising in equilibrium computations of structures, a rapid iterative convergence is hard to obtain. The main reason is that the matrices are structurally symmetric but indefinite, which negatively affects the convergence rate of classical iterative methods. In stability analysis, a straightforward application of some well-known preconditioning techniques, such as incomplete LU factorization, often leads to unacceptable performance. Such preconditioning should be tuned to the problem at hand and used along with an appropriate accelerator.

3.1. Enhanced accelerators

In [16], an adaptive version of a popular solution method GMRES(m) was used for solving difficult structural mechanics problems. Following the results of [15], we consider another version of GMRES(m), deflated GMRES(m) [10, 3]. Deflated GMRES(m) is a variation of

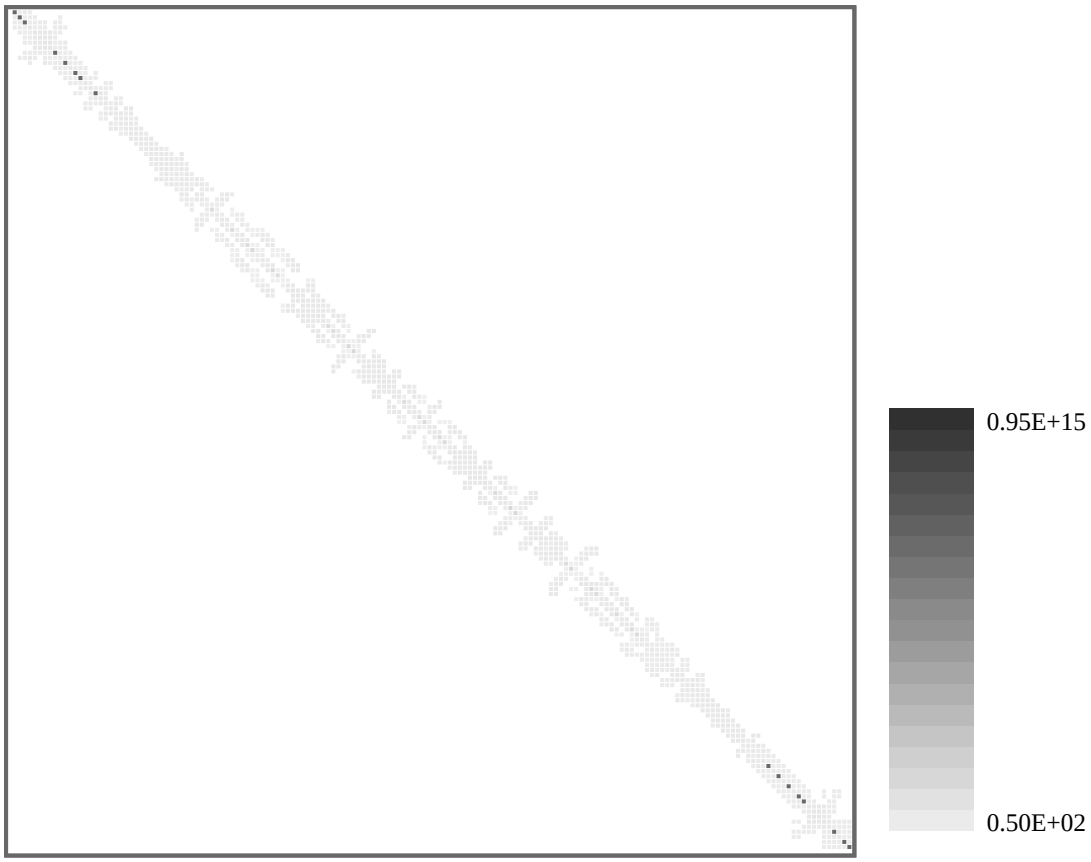

Figure 3. Matrix pattern and values after reordering.

standard GMRES(m) in which eigenvectors corresponding to the smallest eigenvalues are added to the Krylov subspace to prevent stalling of the GMRES(m) convergence. Specifically, eigenvectors from a previous GMRES (outer) iteration are added to the basis of the next Krylov subspace. In addition, this new subspace is also used to improve the accuracy of the desired eigenvalues to be used for the next iteration. Note that the approximate eigenvalues and eigenvectors are obtained as by-products of the iterative process and are inexpensive to compute. At each step, a small generalized eigenvalue problem of size m, the dimension of the Krylov subspace, is solved, see [10, 3] for details.

The gains in performance can be spectacular in some cases. In other cases, specifically when the procedure is not able to capture the smallest eigenvalues accurately, the strategy may not help. In our case, deflation is often quite helpful. For example, for the medium problems in the middle of the nonlinear iteration, deflated GMRES(m) speeds up convergence by 60 iterations compared with restarted GMRES.

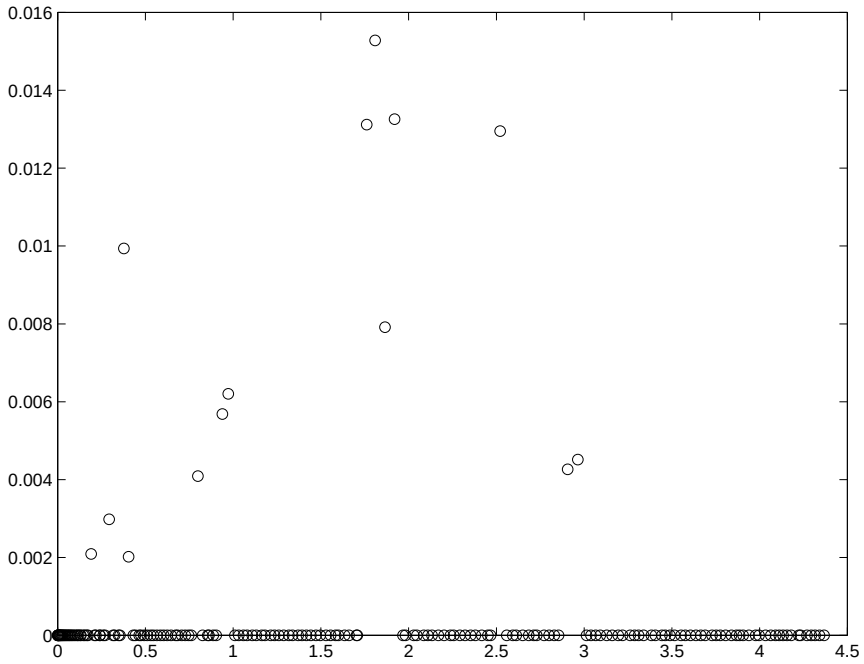

Figure 4. Model $\mathcal{M}$: Estimates for eigenvalues closest to the origin

3.2. Preconditioners

A number of preconditioning options have been tested for the types of problems considered here. Simple diagonal scaling alone or successive overrelaxation (SOR) [12] were not effective. Sparse approximate inverse techniques (see, e.g., [12]) are not considered here because it is difficult to define *a priori* or expensive to capture dynamically a good sparsity pattern for the matrix inverse. Recent developments [2, 5] in these techniques, however, may be effective for the problems considered. Several variations of incomplete LU factorization, such as incomplete LU factorizations with dual threshold (ILUT) and based on levels-of-fill (ILU(k)), both available in SPARSKIT [11], and a block version BILU(k) [4] of ILU(k), were also tested for the problems considered. At first, each of the preconditioning techniques failed to produce a suitable preconditioner. A set of transformations was required to stabilize the preconditioners [15]. When this is done, ILU(k) and ILUT lead to a similar convergence rate for the same number of fill-in elements. Accordingly, this paper will only show the results and transformations of incomplete factorization preconditioners using ILUT as an example.

The first type of preconditioning considered here was constructed based on an incomplete LU factorization in which the fill-in is controlled by a parameter. In particular, a preconditioner ILUT with a dual-threshold dropping strategy was considered (see [12] for a detailed description). A version of ILUT with partial column pivoting (ILUTP) is also available from SPARSKIT and ILUTP was used in the preconditioning to prevent ill-conditioning of the preconditioning matrix.

The second type of preconditioning considered that benefits convergence is the Algebraic

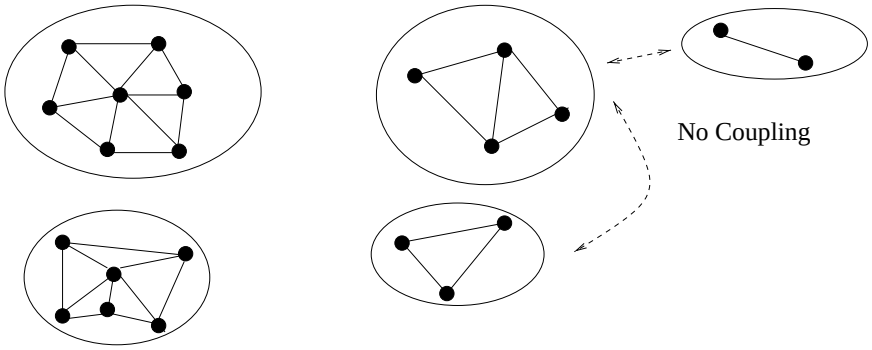

Figure 5. Group independent sets

Recursive Multilevel Solver (ARMS). This is an algebraic multigrid-like algorithm that requires no underlying set of grids for defining prolongation and restriction operators. ARMS starts by reordering the matrix using independent set or group-independent set orderings, see [13] for details. In short, a group-independent set is defined by selecting a set of small 'groups' of nodes in the adjacency graph that are such that nodes between different groups are not coupled. Within each group, nodes can be coupled to each other. This is illustrated in Figure 5.

When reordering the matrix by labeling the nodes of the group-independent set first, a matrix of the following form is obtained

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix},$$

in which B is diagonal or block-diagonal with small blocks.

The above matrix is then approximately block-factored as

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} \approx \begin{pmatrix} L & \\ G & I \end{pmatrix} \begin{pmatrix} U & W \\ & S \end{pmatrix},$$

again using dropping strategies. Then the reordering and factorization are repeated recursively on the Schur complement matrix S, for a small number of levels. At the last level the matrix S is factored using again a standard ILUT or ILUTP factorization. Both the construction of the preconditioner and the forward-backward solutions in ARMS are recursive. In addition ARMS allows inter-level iterations (referred to as W-cycles in the multigrid literature), though these tend to be fairly expensive if the number of levels is high. For more details on this multilevel preconditioner see [14] and [13].

A particular instance of the ARMS preconditioner as well as the ARMS performance for a given iterative algorithm are controlled by several parameters, such as the block size and number of levels specifying the block and level preconditioner structures, respectively. As was mentioned above, ARMS can be set to utilize inner GMRES iterations for solving the Schur complement systems arising at each level (W-cycles). However, in order to use deflated GMRES(m), the preconditioning operation must stay constant in each iteration of deflated GMRES(m). Therefore, no (inner) iterations should be allowed in the levels of ARMS. This can be achieved by setting a proper parameter that selects a construction method in ARMS.

Another important parameter in the ARMS procedure is one which controls diagonal dominance in the B matrix at each level. The algorithm which builds the block-independent set, is programmed to reject the rows that have low diagonal dominance relative to other rows. Specifically, the algorithm builds a vector w consisting of weights which are used for preventing the selection of a given row into the independent set. This is done in two steps. First, some raw diagonal dominance coefficients are computed as

$$\hat{w}(i) = \frac{|a_{ii}|}{\sum_{j=1}^{n} |a_{ij}|}.$$

Note that $0 \leq \hat{w}(i) \leq 1$ and that, when $a_{ii} \neq 0$, the inverse of $\hat{w}(i)$ is $\hat{w}(i)^{-1} = 1 + \sum_{j \neq i} |a_{ij}/a_{ii}|$. Then, from these numbers, a relative set of weights is constructed, for example, from

$$w(i) = \frac{\hat{w}(i)}{\max_{j=1,\ldots,n} \hat{w}(j)}.$$

These weights are now used as a criterion for rejecting rows: if $w(i) < \tau$, then row i will be rejected. For details see [13].

3.3. Adapting preconditioners for tire design problems

The first important observation is that, when tried "as-is" each of the preconditioners we considered failed even for very large fill-in values. In fact, the preconditioner application causes the iterative process to abort immediately indicating a severe instability of the incomplete LU factorizations used. An effective technique to deal with preconditioning instability is to shift the matrix A by a scalar α, construct the preconditioning for $A + \alpha I$, and then apply this preconditioner to the original matrix [9]. The problem, however, is that, for large shifts, the preconditioner becomes an inaccurate approximation of the original matrix inverse. For the linear systems considered here, only rather large shift values (ranging from 0.01 to 0.1) prevented instability of the preconditioner. Combined with pivoting (ILUTP) and with a multilevel approach (ARMS), acceptable preconditioner accuracy and stability could be achieved.

Filtering of small off-diagonal entries was found to be very beneficial in obtaining inexpensive incomplete LU factorization based preconditioners for the shifted matrix $A+\alpha I$. In particular, rows and then columns are scaled by their 2-norms and the matrix is shifted before being filtered. The preconditioner inaccuracy caused by filtering appears to be negligible in this case. However, the preconditioner construction is faster since fewer nonzeros remain in the preconditioner. We have observed that for the problems under consideration after such a filtering process, the majority of (weakly) diagonally dominant rows have *all* their off-diagonal entries dropped. The corresponding rows constitute an independent set, which we call the *trivial independent set*. For the problems under consideration, constructing ARMS with this independent set has the following effects:

(a) The Schur complement submatrix becomes diagonally nondominant.

(b) No dropping occurs in the upper-left submatrix constructed by ARMS.

Table I. Relation between the number of fill-in elements permitted during the preconditioner calculation and the number of iterations till convergence

Problem	`lfil`=100	`lfil`=170	`lfil`=250	`lfil`=400
P1	276	273	273	273
P4	321	320	321	319

(c) After a few reduction levels using trivial independent sets, the most expensive part of the preconditioner construction, the ILUT factorization at the last level, is carried out on a sparser matrix and thus takes much less time than with an arbitrary block independent set.

4. Numerical Experiments

The goal of the experiments reported here is to compare two preconditioning techniques: ILUTP and ARMS. Both preconditioners are constructed for the matrix with shifted diagonal to stabilize the preconditioners. The parameters controlling the amount of fill-in are adjusted such that the total numbers of nonzeros in the resulting preconditioners are approximately the same. Consider symmetric (easy) and nonsymmetric (difficult) matrices in models $\mathcal{M}$ and $\mathcal{L}$ with random initial guess. Deflated GMRES(54) with 4 eigenvectors was used in the experiments to reduce the residual norm by 10^6.

4.1. Varying fill-in

ILUT with pivoting had a dropping tolerance of 10^{-4} and a maximum of 90 and 80 fill-in elements per row in the incomplete LU factors of the model $\mathcal{M}$ and model $\mathcal{L}$ matrices, respectively. When an ILU factorization serves as a preconditioner, a well-known approach to accelerating convergence is an increase of the number of fill-in elements in the preconditioner. However, for large shift values, the accuracy of the preconditioner deteriorates, showing no convergence improvement with larger fill-in. For two linear systems obtained at different steps of the equilibrium computation, Table I presents the dependency of the number of iterations on the parameter `lfil` controlling the number of nonzeros in the ILUT preconditioning with pivoting and shift $\alpha = 0.1$. The relatively low change in the number of iterations with respect to the variation in `lfil` values confirms that the preconditioner will remain a poor approximation of A because of the relatively large shift used, independent of the accuracy used to approximate $A + \alpha I$.

4.2. Varying parameters in ARMS

In ARMS the dropping tolerance of 10^{-2} was used and the last reduced system was factored with an upper bound of 350 fill-in elements. The intermediate levels had a maximum of 250 fill-in elements. In our experiments with ARMS, these upper bounds were never reached, however. On average, the number of nonzeros in the preconditioner was about twice the number of

Table II. Comparison of ARMS for two different block sizes and level numbers

Precon.	bsize	levels	Nonzeros	Construct.	Iterations
A1	10	3	23,424,440	905.80s	213
A2	100	3	20,408,489	2004.65s	324
A3	10	5	23,706,339	1083.56s	217

nonzeros in the original matrix. The upper bounds on fill-in and the dropping tolerance were selected such that the construction of the ARMS preconditioner is not the major cost and the preconditioner is sufficiently well-conditioned to enable an acceptable convergence rate.

Our experience with this type of problems shows that taking small blocks (of size 3 or 10) instead of larger blocks (of size 100) yields a better ARMS preconditioner. This result might be attributed to the block structure of the original matrices. For example, Table II compares three variations of the ARMS preconditioner for a symmetric model $\mathcal{M}$ problem. Two of them, A1 and A2, have the same values of all the parameters except for the block size, which is equal to 10 and 100 (column `bsize`), respectively. The total preconditioner construction time is shown in the column `Construct`, the number of iterations until convergence is in the column `Iterations`, and the number of nonzeros in preconditioner is given in the column `Nonzeros`. Note that no filtering or trivial independent set was used to obtain results in Table II.

Varying the number of ARMS levels from 2 to 5 did not affect significantly the preconditioner performance, but fewer levels made the preconditioner construction less expensive. In Table II, preconditioners A1 and A3 are constructed with the same values for all the parameters but the number of levels (column `levels`), is 3 and 5, respectively. The number of levels was set to two for the experiments with ARMS in the next subsections.

4.3. Block Independent Set strategies and filtering the matrix

Filtering of the small (10^{-3}) off-diagonal entries has been performed to construct the preconditioner. For a symmetric model $\mathcal{M}$ matrix, Table III shows the dependence between the complexity of block independent sets in ARMS (controlled by the parameter $w(i)$, $i = 1, \ldots, n$, defined to be the relative diagonal dominance of matrix row i, see subsection 3.2) and the numbers of nonzeros in the entire ARMS preconditioner and in its last level (solved by ILUT). The total preconditioner construction time is shown in the column `Construct`. As indicated by the data in the column `Iterations`, a more positive effect on convergence is seen when trivial independent sets are used (i.e., when $w(i) = 1.0$ for all the rows) in preconditioner construction. This can be attributed to a better conditioning of the B block in the case of trivial independent sets. The columns `Last System` and `Total` present the number of nonzeros in the linear system produced at the last (third) level of the block factorization and in the entire preconditioner, respectively. When trivial independent sets are used, the last-level linear system is often quite large, since larger values of the parameter w usually generate smaller independent sets rejecting more rows as not sufficiently diagonally dominant. With $w = 1.0$, the last reduced system is also sparser, which makes the preconditioner construction less expensive. In general, it seems beneficial to use trivial independent sets when, for other values of the parameter $w(i)$, $i = 1, \ldots, n$ in ARMS, the size of the reduced system remains large

Table III. The effect of varying the independent set parameter on the preconditioner

w	Construct.	Iterations	Last System	Total
0.1	278.96s	130	12,275,196	14,104,476
1.0	66.13s	98	7,932,632	8,026,226

relative to the size of the original one. For the ILUTP preconditioner, if we reorder the input matrix such that the rows in the trivial independent set represent its upper-left submatrix, the resulting preconditioner does not lead to convergence within 2,000 iterations. This suggests that a multilevel approach may play a major role in accelerating convergence.

4.4. Performance variation with respect to right-hand side

It has been observed that the choice of the right-hand side may affect convergence significantly. If the right-hand side is taken at random, for example, in the solution of a symmetric model $\mathcal{M}$ problem, then the residual norm is reduced only by 10^5 when ARMS is applied in 2,000 iterations, and the convergence stagnates with ILUTP. On the other hand, if the right-hand side is chosen such that the solution consists of all ones (Figures 6 and 7) or if the right-hand side is obtained from the physical model and properly scaled (Figures 8 and 9), then the convergence is not hindered. The latter two figures display almost identical convergence histories for the ILUTP and ARMS preconditioners: both rapidly reduce the residual in the beginning and tail off towards the end of the convergence. It takes much less time, however, to construct the ARMS preconditioner (see Tables IV and V) because an incomplete ILUT factorization, which is quite expensive, is performed on a smaller and sparser system only in the last level of ARMS.

In Tables IV and V, the solution times (on a DEC Alpha) are shown for the convergence curves in Figures 6, 7 and Figures 8, 9, respectively. From Table IV, it is clear that both methods work harder for the system at the end of the nonlinear Newton iteration when all the nonlinear constraints are enforced. Note that the ARMS preconditioner outperforms ILUTP for the difficult problems (Figure 7, right and Figure 8, right). It appears in this case that the high degree of nonsymmetry and the low rate of diagonal dominance do not have as negative an effect on ARMS as they do on ILUTP.

4.5. The curse of poor conditioning

In all cases, convergence slows down considerably after the residual norm is reduced by a factor of about 10^{-4}. (The same residual norm reduction was observed when the residual was computed by matrix-vector multiplication or by the linear combination of the search directions.) At this stage of the convergence history, there is almost no reduction in the error norm of the solution, Typically, a low-accuracy solution can be effectively used in a nonlinear iteration performed with an inexact Newton-type method (see, e.g., [7]). However, our attempts to employ the obtained linear system solution in the context of these nonlinear methods were not successful due to the lack of solution accuracy. Thus, a higher accuracy is desired from the iterative process for the solution to be usable even by the inexact Newton

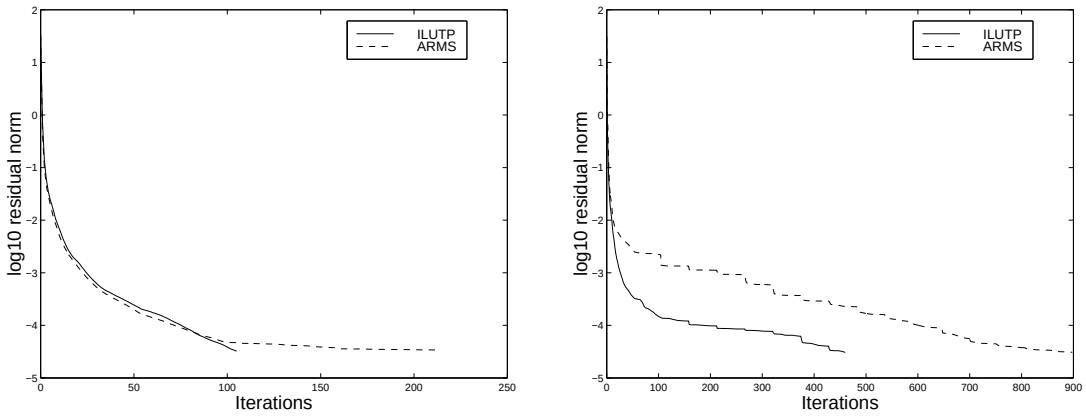

Figure 6. Model $\mathcal{M}$: Residual norm reduction for symmetric (left) and nonsymmetric (right) linear systems with right-hand side corresponding to the solution of all ones

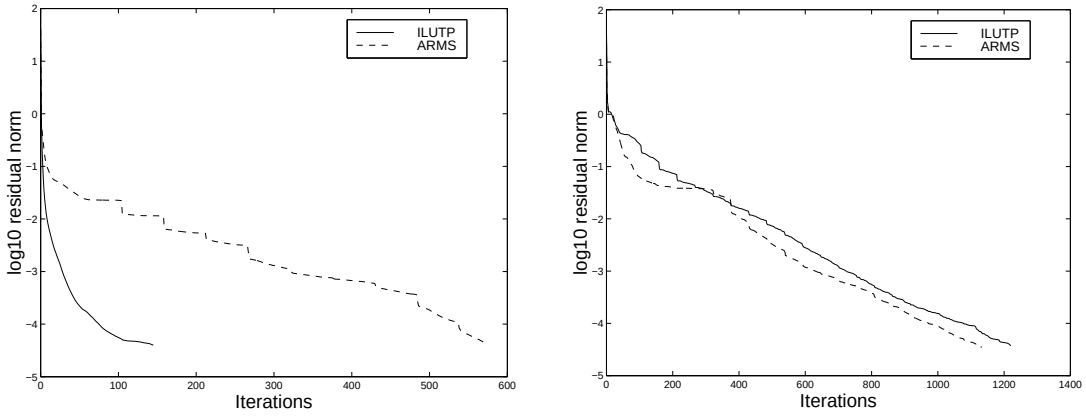

Figure 7. Model $\mathcal{L}$: Residual norm reduction for symmetric (left) and nonsymmetric (right) linear systems with right-hand side corresponding to the solution of all ones

methods. To accelerate the convergence, an iterative refinement would be beneficial for a reasonably conditioned system, for which stagnation occurs at the level of working precision. In [17], it has been shown that evaluating the residual in higher precision allows greater residual norm reduction in the working precision. The problems under consideration are highly ill-conditioned, however, and iterative refinement produced no significant improvement in the convergence compared with continuing to iterate with the working precision. In particular, when the residual is evaluated in higher precision at each restart of deflated GMRES, which is run for several restart cycles, there is no improvement in the residual norm reduction. As was already mentioned, the linear systems tend to become harder to solve as the nonlinear iterate converges to the solution.

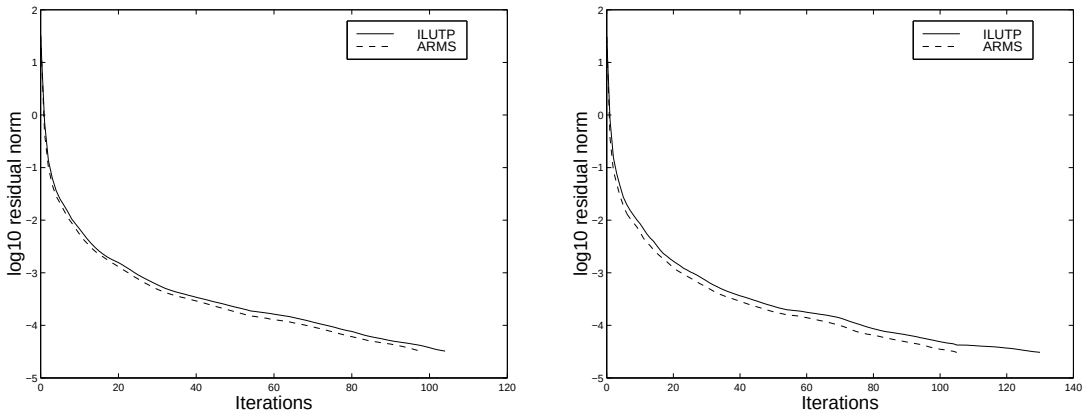

Figure 8. Model $\mathcal{M}$: Residual norm reduction for symmetric (left) and nonsymmetric (right) linear systems with right-hand side taken from the physical model

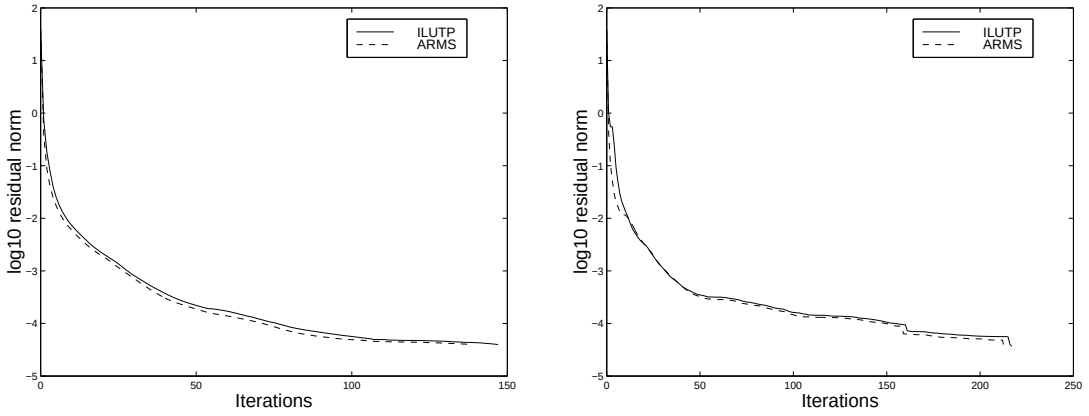

Figure 9. Model $\mathcal{L}$: Residual norm reduction for symmetric (left) and nonsymmetric (right) linear systems with right-hand side taken from the physical model

4.6. The nonlinear context

To show the varying difficulty of the linear systems in a nonlinear iteration, a complete sequence of systems has been solved using the ARMS preconditioner. This sequence corresponds to a symmetric nonlinear system of 20,125 unknowns with the matrix characteristics similar to symmetric model $\mathcal{M}$ and $\mathcal{L}$ problems. Solution times for each of the eight linear systems are depicted in Figure 10.

5. Conclusions

The sparse linear systems arising in tire equilibrium computations are challenging for iterative solution methods. The treatment of stationary solutions of rotation makes the systems

Table IV. Solution and preconditioner construction CPU times (in seconds) for symmetric and nonsymmetric problems with right-hand side corresponding to the solution of all ones

	model $\mathcal{M}$				**model $\mathcal{L}$**			
	Symmetric		Nonsymmetric		Symmetric		Nonsymmetric	
Precon.	ILUTP	ARMS	ILUTP	ARMS	ILUTP	ARMS	ILUTP	ARMS
Construct.	404.09	66.08	408.60	74.26	727.53	122.22	643.58	115.72
Solution	114.39	228.75	518.49	1003.95	256.60	1056.70	2206.18	2126.07
Total	518.48	294.83	927.09	1078.21	984.13	1178.92	2849.76	2241.79

Table V. Solution and preconditioner construction CPU times (in seconds) for symmetric and nonsymmetric problems with right-hand side taken from the physical model

	model $\mathcal{M}$				**model $\mathcal{L}$**			
	Symmetric		Nonsymmetric		Symmetric		Nonsymmetric	
Precon.	ILUTP	ARMS	ILUTP	ARMS	ILUTP	ARMS	ILUTP	ARMS
Construct.	402.64	66.13	408.34	73.69	728.37	122.30	644.58	116.50
Solution	113.12	102.41	144.73	112.83	256.83	250.72	381.09	394.14
Total	515.76	168.54	553.07	186.52	985.20	373.02	1025.67	510.64

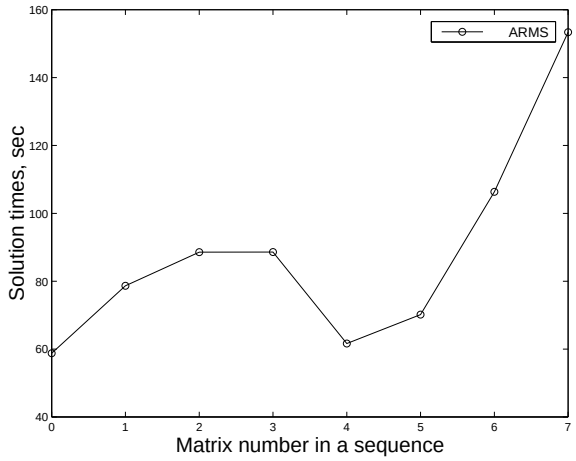

Figure 10. Solution times for the sequence of eight linear systems

extremely ill-conditioned during the nonlinear convergence process. In such a situation, very good preconditioning is mandatory. However, a large number of diagonally nondominant rows and columns hinders the effectiveness of preconditioning for accelerating iterative method convergence.

The performance of a new multilevel preconditioning technique (ARMS) has been compared with a variation of an incomplete LU factorization (ILUTP) that benefits convergence for this class of problems in structural mechanics. Both ILUTP and ARMS have been combined with a deflated version of GMRES(m) and augmenting the diagonal elements. An unexpected observation is that for both preconditioners convergence is improved *only* when a rather large shift is applied to the matrix diagonal. As a result, increasing the allowed fill-in in both preconditioners does *not* reduce the number of iterations for convergence. In summary, the main conclusions from the comparison are: (1) the multilevel approach is more advantageous for difficult nonsymmetric problems whereas an ILU preconditioner delivers comparable or better performance for symmetric problems. (2) Shifting of the diagonal leads to a more stable preconditioner for both ILUTP and ARMS factorizations. The size of the shift is very important: while making the preconditioner more stable, large shift values cause the preconditioner to be a poor approximation of the original matrix. (3) Unless special precautions are taken, the preconditioner construction may easily dominate the solution cost. Special techniques such as filtering of small entries and the use of trivial independent sets (in ARMS) have been found to be very helpful in reducing the cost of the preconditioner construction. (4) The solution of linear systems arising in tire design depends heavily on the right-hand side value. (5) Both preconditioners are rather robust since they enable the solution of the linear systems of varying difficulty with the relative tolerance of 10^{-6} in the residual norm reduction. A higher accuracy in the solution has not been achieved. It is conceivable that extreme ill-conditioning prevents one from obtaining further residual reduction unless a very accurate preconditioner, close to a direct solver, is used. This remains a subject of further investigation.

ACKNOWLEDGEMENT

We are grateful to Brian Suchomel for the helpful suggestions regarding the experiments with the multilevel preconditioner.

REFERENCES

1. K.J. Bathe and E.L. Wilson. *Numerical Methods in Finite Element Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.
2. M. Benzi, J.C. Haws, and M. Tuma. Preconditioning highly indefinite and nonsymmetric matrices. Technical Report LA-UR-99-4857, Los Alamos National Laboratory, Los Alamos, NM, August 1999.
3. A. Chapman and Y. Saad. Deflated and Augmented Krylov Subspace Techniques. *Numer. Linear Algebra Appl.*, Vol. 4:pages 43–66, 1997.
4. A. Chapman, Y. Saad, and L. Wington. High–order ILU Preconditioners for CFD Problems. Technical Report UMSI 96/14, Supercomputer Institute, Univ. Minnesota, 1200 S. Washington Ave., Minneapolis, MN 55415, 1996.
5. E. Chow. A priori sparsity patterns for parallel sparse approximate inverse preconditioners. Technical Report UCRL-JC-130719 Rev.1, Lawrence Livermore National Laboratory, Livermore, CA, 1999.
6. J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Anal. Appl.*, 20:720–755, 1999.

7. S. C. Eisenstat and H. F. Walker. Globally convergent inexact newton methods. *SIAM J. Optimization*, 4:393–422, 1994.
8. P. Le Tallec and C. Rahier. Numerical Methods of Steady Rolling for Non–Linear Viscoelastic Structures in Finite Deformations. *Int. J. Numer. Methods Eng.*, Vol. 37:pages 1159–1186, 1994.
9. T.A Manteuffel. An incomplete factorization technique for positive definite linear systems. *Mathematics of computation*, 32:473–497, 1980.
10. R.B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.*, Vol. 16:pages 1154–1171, 1995.
11. Y. Saad. SPARSKIT: A basic toolkit for sparse matrix computations. Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990.
12. Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing, New York, 1996.
13. Y. Saad and B. Suchomel. ARMS: An algebraic recursive multilevel solver for general sparse linear systems. Technical Report umsi-99-107, University of Minnesota Supercomputer Institute, Minneapolis, MN 55415, 1999.
14. Y. Saad and S. Zhang. BILUTM: A Domain-Based Multi-Level Block ILUT Preconditioner for General Sparse Matrices. Technical Report UMSI 98/118, Supercomputer Institute, Univ. Minnesota, 1998.
15. M. Sosonkina, J. Melson, and L.T. Watson. Iterative Solution of Large Linear Systems Arising in Tire Design. In S.N. Atluri and P.E. Donoghue, editors, *Modeling and Simulation Based Engineering*, volume 1, pages 473–478. Tech Science Press, 1998.
16. M. Sosonkina, L.T. Watson, R.K. Kapania, and H.F. Walker. A New Adaptive GMRES Algorithm for Achieving High Accuracy. *Numer. Linear Algebra Appl.*, Vol. 5:pages 275–297, 1998.
17. K Turner and H. F. Walker. Efficient high accuracy solutions with GMRES(m). *SIAM J. Sci. Stat. Comput.*, 13:815–825, 1992.